Computing and Software System Design Description: SDP

020.50.15.00.00-0001 DSN

Status: **RELEASED**

PREPARED BY	ORGANIZATION	SIGNATURE
M. Whitehead, DMS Software Architect	DMS, NRAO	Signed by: Mark Whitehead FAC39E16435B4FE... 10/16/2025

APPROVALS	ORGANIZATION	SIGNATURES
R. Rosen, CSS IPT Lead	ngVLA, NRAO	Signed by: Rachel Rosen 4602AA3F0A7C428... 10/16/2025
E. Murphy, Project Scientist	ngVLA, NRAO	Signed by: Eric Murphy 82D35C8655C14D1... 10/16/2025
R. Selina, Project Engineer	ngVLA, NRAO	Signed by: R. Selina 9740299D489B4DD... 10/16/2025
P. Kotzé, System Engineer	ngVLA, NRAO	Signed by: P.P.A. Kotzé 9E6E22E5C0FB448... 10/16/2025

RELEASED BY	ORGANIZATION	SIGNATURE
W. Hojnowski Project Manager	ngVLA, NRAO	Signed by: William Hojnowski 722BEBEC6DE84DB... 10/16/2025

Change Record

Version	Date	Author	Affected Section(s)	Reason
1	2025-08-11	M. Whitehead	All	Initial draft
A	2025-08-13	M. Archuleta	All	Minor edits and formatting for pdf and release.
B	2025-10-07	M. Whitehead	1.2 (RID482), 8.3 (RID480), 8.3 (RID475), 6.5 (RID477)	Post-CDR edits: RID482 Identify “package” RID480 pybind RID475 Infrastructure Management RID477 “Container” nomenclature

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This document describes the ngVLA Science Data Processor (SDP) design constraints, architecturally significant requirements, conceptual design, and plan for evolving the design to the Preliminary Design Review (PDR). SDP is part of the ngVLA Computing and Software System (CSS) shown in Figure 1.

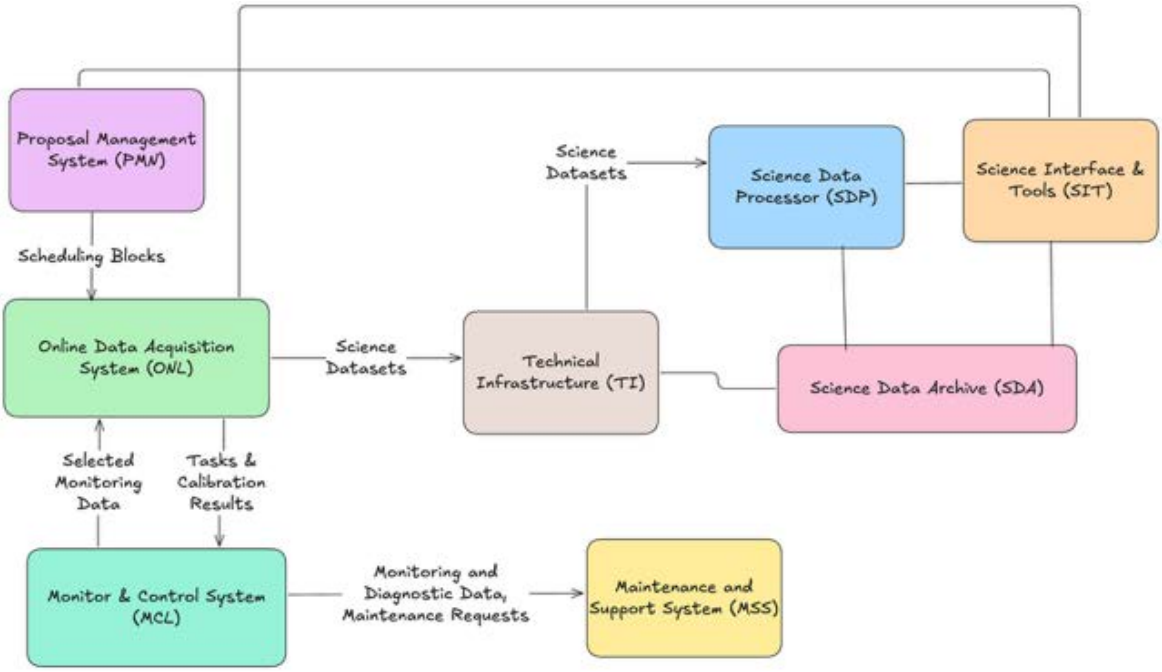

Figure 1. General overview of the ngVLA Computing and Software System (CSS).

1.2 Scope

The Radio Astronomy Data Processing Program will develop a data processing system that supports production of high-level data products for the Atacama Large Millimeter/submillimeter Array (ALMA) Wideband Sensitivity Upgrade (WSU) and the next generation Very Large Array (ngVLA). As a secondary objective, the system will support the continued evolution of radio data astronomy processing through a widely accessible package that enables specialization and innovation at facilities and universities. "Package" refers to a cohesive set of software that provides users or system components with the necessary applications, libraries, and utilities to perform specific data-related functions - such as visibility processing, data processing, or data analysis - either within SDP or on user-provided computing resources.

This document describes the conceptual data processing design applicable to both ALMA WSU and ngVLA in the context of the ngVLA systems architecture as an observatory system. This document does not include ALMA-Data Processing, Data Management, and Data Flow System (ALMA-D3) context nor does it directly address needs associated with an accessible data processing package that enables specialization and innovation at facilities and universities.

In this document, Radio Astronomy Processing System (RADPS) and Science Data Processor (SDP) are synonymous.

1.3 Software Architecture Model

In this document, C4 is used to model the SDP software architecture[1]. C4 is a lean, developer-friendly approach to software architecture diagramming that reflects how software architects and developers think about software. C4 elements include:

- Context: The system that is modeled.
- Container: In C4, a container is not a Docker container. Instead, a container is any process or data store that needs to be running for the overall system to work.
- Component: A grouping of functionality encapsulated behind a well-defined interface.
- Code: Basic building blocks of the programming language; classes, interfaces, functions, etc.

2 Reference Documents

The following documents are referenced within this text:

Ref. No.	Document Title	Rev/Doc. No.
RD01	Computing and Software Systems Use Cases	020.50.00.00.00-0006 USC
RD02	Radio Astronomy Data Processing System User Interfaces (RADPS-UIs), Concept of Operations, v0.3	
RD03	NA ALMA Development Proposal – RADPS-ALMA: A Data Processing System for ALMA WSU.	
RD04	System Architecture Description	020.10.20.00.00-0002 REP
RD05	Imaging Unmitigated ALMA Cubes	NAASC Memo #121
RD06	Domain Infrastructure Prototype	
RD07	System Concept Options and Trade-offs	020.10.25.00.00-0005 REP
RD08	Software Development Roadmap	020.50.00.00.00-0009 PLA
RD09	ngVLA Data Rates and Computational Loads (Update)	ngVLA Computing Memo 11
RD10	System Considerations for ngVLA Data Processing, Transport and Storage Systems	ngVLA Computing Memo 12
RD11	Algorithm Architecture	
RD12	Dask Centered ngData Processing Architecture and VIPER Software Framework	
RD13	An Example RADPS Workflow Decomposition	RADPS-006

[1] http://c4model.com

3 System Overview

Figure 2 shows the ngVLA system context from the perspective of the SDP. Connecting lines represent generic relationships.

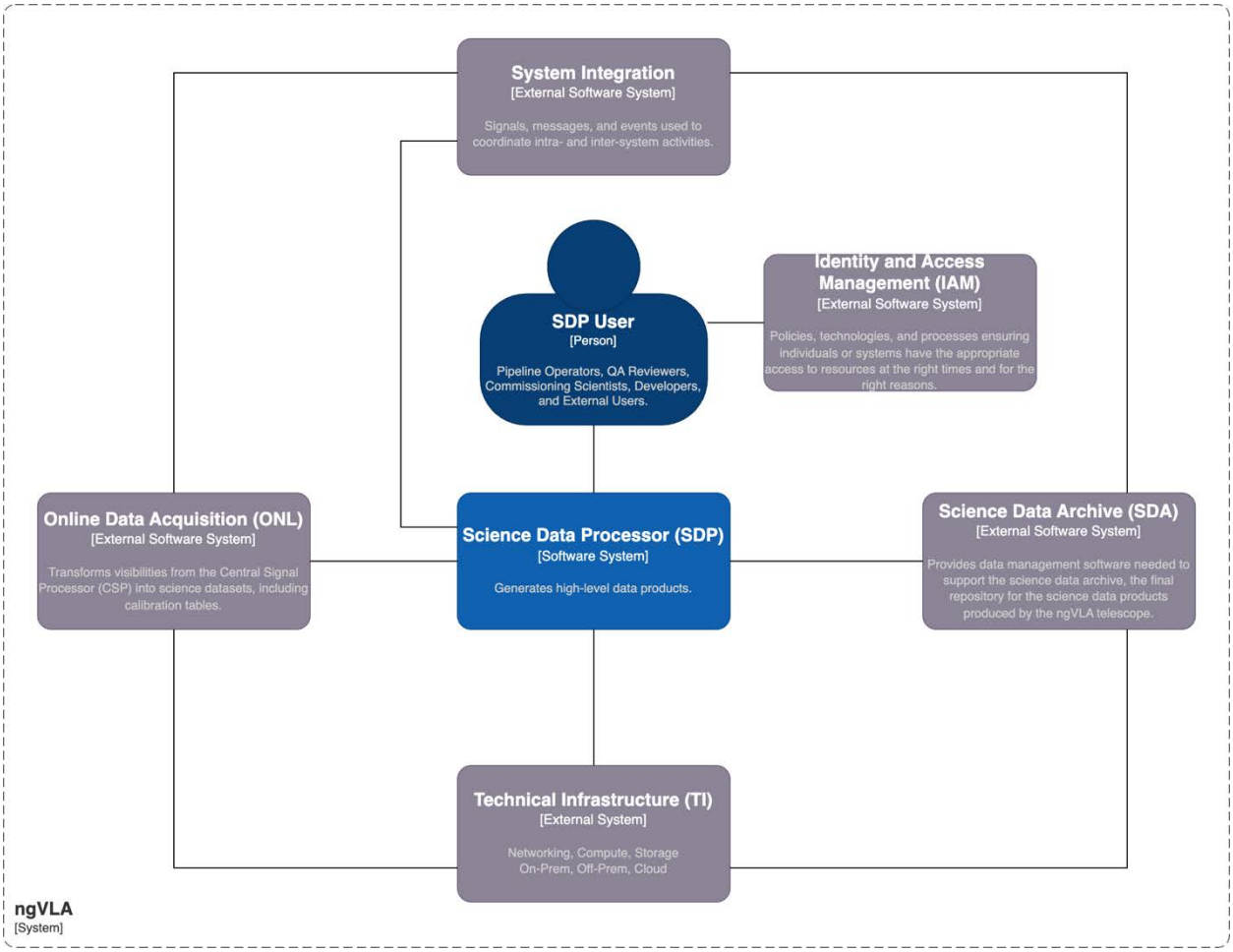

Figure 2. Science Data Processor Context View.

The SDP is responsible for all data processing functions that occur after data acquisition, online calibration table generation, online radio frequency interference (RFI) mitigation, and science dataset creation. These functions include the generation of high-level data products and providing quality assessment.

The SDP is designed for a broad set of users involved in processing, reviewing, and analyzing astronomical data. These include internal roles such as pipeline operators, quality assessment reviewers, commissioning scientists, and software developers, as well as external scientists who use the processed data in their research. Each group has a different focus: pipeline operators manage the flow and status of processing; quality assessment reviewers evaluate data quality and make processing adjustments;

commissioning scientists test and refine processing methods; developers design and troubleshoot the software itself; and external scientists rely on the system for accurate, usable data products.
To support these users, SDP offers a set of standard browser-based views tailored to common workflows. These include interfaces for managing large-scale processing tasks, reviewing individual projects, and analyzing data quality. Together, these standard views form a cohesive framework that supports day-to-day operations, allowing users to access, monitor, and interact with data efficiently.
SDP critically depends on Technical Infrastructure (TI), which encompasses all computing, storage, and networking hardware as well as logging and monitoring systems and data transport and communication middleware.

4 Constraints

Table 1 summarizes the technical and non-technical constraints on the SDP architecture.

Table 1. SDP Conceptual Design Constraints.

ID	Description	Reference
CON1	The RADPS conceptual design must address the requirements for both ngVLA and ALMA/WSU.	RD03
CON2	The system will archive visibilities before asynchronously generating high-level data products.	RD07
CON3	Data reduction will occur on-premises in a Science and Operations Data Center as well as off-premises in partner facilities (e.g., Texas Advanced Computing Center or Center for High-Throughput Computing) or in the cloud (e.g., AWS, GCP, etc.).	RD07
CON4	DMS is adopting Agile to support incremental design and implementation.	RD08
CON5	DMS staff need to be engaged in the conceptual and preliminary design phases.	RD08
CON6	DMS staff possess limited large-scale, distributed data processing expertise.	RD08
CON7	Python is the lingua franca in the astronomy domain. DMS staff have considerable experience primarily with Python and secondarily with C++.	N/A

CON2 requires SDP to be a batch processing system. That is, the system will process finite and static datasets.
CON4 and CON 5 imply the architecture needs to support creating a "skeletal" system with minimal functionality to exercise execution pathways and establish performance and reliability baselines. DMS teams will iteratively add functionality and measure the system's important quality attributes.
To address CON6, DMS staff will focus on areas of NRAO domain expertise and engage large-scale, distributed data processing experts to produce Technical Infrastructure design products. To increase technical literacy in this area, the engagement will include knowledge transfer to DMS staff.
CON7 drives the design to favor Python-based technologies. Additionally, DMS has demonstrated that passing data from Python processes to C++ libraries through pybind11 does not impact performance

[RD06], so the design should be able to incorporate C++ libraries as needed to satisfy important quality attributes.

5 Architecturally Significant Requirements

An Architecturally Significant Requirement (ASR) is a requirement that has a large impact on the architecture such that changing the requirement would significantly change the architecture. A quality attribute (QA) is a measurable or testable property of a system used to indicate how well it satisfies stakeholder needs.
The general approach for creating the SDP conceptual architecture includes identifying ASRs, organizing ASRs by quality attributes, deriving architecture decisions from ASRs, and using those decisions along with constraints to decompose SDP into subsystems and other entities.
The subsection headers below provide the quality attributes, subsection tables summarize related ASRs, and subsection text summarizes architecture decisions.

5.1 Performance

Table 2. Performance ASRs.

ID	Name	Text
CSS9512	Continuum Images of Triggered Observations	The Science Data Processor shall produce continuum images for triggered observations of 1 hour duration no more than 1 hour upon completion of the observation.
CSS9512.1	Quick Look Image Pipeline Products	For triggered observations, there shall be a standard data reduction performed resulting in a continuum image, processed in a time duration equal to or less than the observation duration. The computational load generated for the triggered observation shall not exceed 60 PFLOPs/sec.
CSS9600	Computing throughput required for synthesis imaging	The data processing system shall be able to sustain an interferometric average throughput of 60 PFLOPS and peak throughput of 1354 PFLOPS.
ALMA-TR56	Automatic continuous calibration of sessions	The DPS shall have the ability to automatically complete calibration processing of a session no later than 12 hours after completion of a session on the telescope and to maintain this processing pace for all new data coming from the telescope. Re-calibration, if required upon completion of antenna positions, shall also be automatically initiated, and DPS and Archive shall have the capacity to simultaneously keep up with that reprocessing.
ALMA-TR59	Total power image generation	When the total power portion of a GOUS has completed observing, the DPS should do baseline subtraction and total power image generation and store the results in the Archive in less than three days.
ALMA-TR65	Producing GOUS level image products	When all the data in a GOUS have been calibrated, the DPS should automatically produce the GOUS level image

ID	Name	Text
		products and ingest them into the Archive in less than 10 days, including any additional processing steps specific to GOUS (e.g. aligning the astrometry for different MOUSes, re-scaling Solar data).

A combined strategy of horizontal and vertical scaling, along with data parallelism, is the only feasible option to satisfy the throughput and time requirements listed in Table 2. Horizontal scaling involves designing and managing systems in a way that allows them to distribute workloads across multiple nodes and handle increasing workloads by adding identical resources. Vertical scaling complements this by enhancing the capabilities of individual nodes through increased memory, CPU cores, or network bandwidth.

Data parallelism underpins both strategies by dividing large datasets into independent partitions that can be processed simultaneously, ensuring high throughput and reduced latency. This model is particularly effective when the processing tasks are uniform and stateless, enabling straightforward replication across compute resources.

Architecture Decision: Horizontal and vertical scalability in conjunction with appropriate data parallelism is the only feasible approach to address throughput and timing performance requirements.

5.2 Reliability

Table 3. Reliability ASRs.

ID	Name	Text
CSS9608	Hardware Fault Tolerance	The data processing system shall tolerate failures in the hardware elements (computing nodes, storage nodes, etc.) involved in an execution without losing the data computed for the observation so far. For gridding, setting the boundary of successful completion at scan (or scan-slice) level should be sufficient. Other operations may require other strategies (e.g. checkpointing).
ALMA-TR39	Failed processes	If the processing fails for any reason, the failing process shall not corrupt the data, shall be automatically restarted if the failure is most likely intermittent, and shall save diagnostic information for a post facto determination of the cause.

In large-scale, distributed data processing systems, where failures are statistically inevitable due to the scale and duration of operations, restarting entire workflows from scratch is impractical and inefficient. Rollback recovery, a fault-tolerance technique that restores a system to a previously known good state so that it can resume execution as if the failure had not occurred, is one of the most widely used fault tolerance techniques for high performance computing systems[2] and is critical for ensuring that long-running computations can be restarted from the point of failure without reprocessing completed work. Checkpointing, a mechanism used to enable rollback recovery, involves periodically capturing the state of a computation which can be used to resume execution after a failure. This approach enables the

[2] Dongarra et al., "Fault Tolerance Techniques for high-performance computing", Springer Verlag, 2015

system to maintain forward progress despite hardware faults, network interruptions, or software crashes.

Architecture Decision: Rollback recovery via checkpointing is one of the most widely used fault tolerance techniques for high performance computing systems and is currently the most feasible option for ensuring that long-running computations can be restarted from the point of failure without reprocessing completed work.

5.3 Maintainability

Table 4. Maintainability ASR.

ID	Name	Text
CSS7018	Design Life	The system shall be designed for an expected operational life of no less than 20 years, where the operational life is defined to start at the full operations milestone and close-out of the construction project.

The software engineering principle of separation of concerns is essential to ensure the long-term maintainability of a large-scale, distributed data processing system intended to operate for at least twenty years. By decomposing SDP into distinct, well-defined elements - each responsible for a specific function such as workflow orchestration or compute execution - the architecture becomes more modular, understandable, and adaptable.

This modularity enables individual elements to be updated, replaced, or scaled independently without triggering large changes to the code base. It also facilitates clear interfaces, better testing practices, and focused documentation, all of which are important for preserving institutional knowledge over time as personnel and technologies change.

Architecture Decision: The SDP conceptual design must realize a strict separation of concerns so that SDP remains extensible for emerging technologies and new requirements and resilient across decades of scientific and operational change.

5.4 Reusability

Table 5. Reusability ASRs.

ID	Name	Text
CSS9502.1	Non-Observatory Resource Allocation	The Science Data Processor shall be able to allocate resources on non-Observatory computing facilities.
CSS10002	Tools for Visibility Processing – Supported Platforms	A data processing tool kit shall be provided for users to generate high-level data products for non-standard modes using user-provided computing resources that conform to observatory-defined standards. An observatory data processing platform standard shall be defined, encompassing the following deployment types: in-house computing resources, dedicated contributed computing resources, surge computing resources, collaborative computing resources, personal user computing resources.

ID	Name	Text
ALMA-TR38	Allow incorporation of additional computing resources	The DPS shall allow all the incorporation of additional computing resources, either in the form of on-demand resources or additional permanent resources, without disrupting the data processing flow.
ALMA-TR70	User calibration, visibility manipulation, and imaging on user own compute systems.	Users shall be able to calibrate visibility data, perform basic mathematical manipulation of visibilities and image visibility data using a high-level application interface, and the same heuristics and algorithms as the production system, on their own compute systems, subject to the processing limitations of the system.

The software engineering principle of separation of concerns is necessary to support reusability of radio astronomy algorithms across a diverse range of computing environments including in-house systems, dedicated high-performance computing (HPC) facilities, cloud-based surge capacity, university clusters, and personal user machines.
Cleanly decoupling algorithm logic from infrastructure-specific concerns such as job submission mechanisms, data staging strategies, or monitoring tools means those concerns can evolve independently of the core algorithm implementations and algorithms can remain environment-agnostic and portable. Containers provide a lightweight, standardized environment that bundles all necessary dependencies, libraries, and configurations, allowing the same algorithm to run identically across diverse systems without modification. This approach simplifies deployment, reduces environment-specific errors, and ensures consistent behavior across development, testing, and production settings. Separation of concerns further enables infrastructure-specific orchestration to be managed independently from the scientific code.
Architecture Decision: The SDP conceptual design must enforce a strict separation of concerns between core radio astronomy domain algorithms and the underlying Technical Infrastructure. By decoupling algorithm logic from infrastructure-specific dependencies and encapsulating runtime environments within containers, algorithms can be made portable, consistent, and reproducible regardless of the underlying Technical Infrastructure.

5.5 Multitenancy

Table 6. Multitenancy ASRs.

ID	Name	Text
RD02	*Radio Astronomy Data Processing System User Interfaces (RADPS-UIs), Concept of Operations, v0.3*	Five user types (i.e., Pipeline Operator, Quality Assessment Reviewer, Developer, Commissioning Scientist, and External User)
RD01	*Preliminary System Architecture Description*	Different users will have various levels of access to the system and the system will simultaneously run prioritized jobs.
ALMA-TR40	Processing priorities	The DPS shall have the ability to assign different processing priorities to different jobs.

ALMA-TR107	Meta-data driven processing	For supported pipeline-able modes, the DPS shall determine how to process the input data using the related meta-data information and the data itself with no required human input.

Multitenancy enables multiple user groups to securely share the same underlying Technical Infrastructure - whether in-house clusters, HPC centers, cloud platforms, or university systems - while maintaining isolation of their workflows, data, and configurations. Adopting a multitenant architecture is necessary to efficiently manage and execute user group requests for data processing across a diverse range of computing environments.

This model supports efficient resource utilization by allowing dynamic allocation of compute and storage resources based on demand or priority, reducing idle capacity, and enabling parallel operations across user groups. It also simplifies system administration through centralized management of common services, while enabling customizable access controls, quotas, and environments tailored to each group's needs.

Architecture Decision: The SDP conceptual design must incorporate multitenancy to enhance scalability, operational efficiency, and collaboration and support a broad and evolving community of users.

5.6 Architecture Decision Summary

Table 7. Architecture Decision Summary.

ID	Quality Attribute	Description
AD01	Performance	Horizontal and vertical scaling in conjunction with appropriate data parallelism is the only feasible approach to address throughput and timing requirements.
AD02	Performance	CON2 requires SDP to be a batch processing system. That is, the system will process finite and static datasets.
AD03	Reliability	Rollback recovery is one of the most widely used fault tolerance techniques for high performance computing systems and is currently the most feasible option for ensuring that long-running computations can be restarted from the point of failure without reprocessing completed work.
AD04	Maintainability	The SDP conceptual design must realize a strict separation of concerns so that SDP remains extensible for emerging technologies and new requirements and resilient across decades of scientific and operational change.
AD05	Reusability	The SDP conceptual design must enforce a strict separation of concerns between core radio astronomy domain algorithms and the underlying Technical Infrastructure. By decoupling algorithm logic from infrastructure-specific dependencies and encapsulating runtime environments within containers, algorithms can be made portable, consistent, and reproducible regardless of the underlying Technical Infrastructure.

ID	Quality Attribute	Description
AD06	Multitenancy	The SDP conceptual design must incorporate multitenancy to provide scalability, operational efficiency, and collaboration across a broad and evolving user community.

6 Conceptual Architecture

Applying AD04, general separation of concerns, with respect to the other architecture decisions produces the SDP system decomposition shown in Figure 3. The boxes represent concepts, and the lines represent generic relationships.

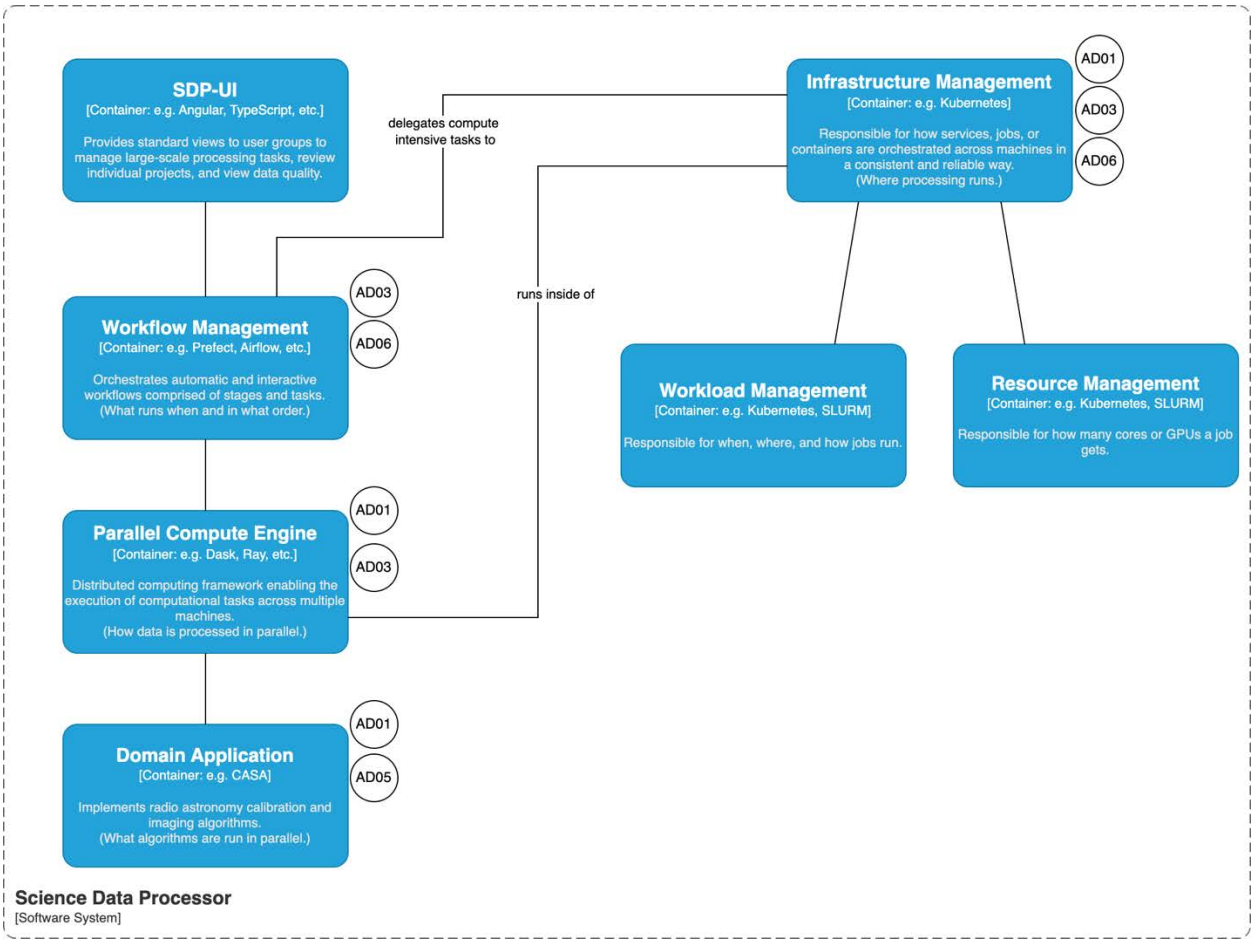

Figure 3. SDP Subsystem Container View.

6.1 Workflow Management

A workflow is a structured sequence of tasks that defines how data or operations are processed over time. Each task represents a discrete unit of work such as data extraction, transformation, computation, or storage and the workflow specifies their order, dependencies, and conditional logic. Workflows can

be considered multi-stage pipelines that can span different systems and enable data processing automation and scalability.
Structurally, the Workflow Management subsystem provides an interface to available workflows for multiple user groups. Behaviorally, the subsystem enables multiple user groups to automatically or interactively execute workflows and orchestrates the execution of multiple workflows.

6.2 Parallel Compute Engine

The Parallel Compute Engine represents a distributed computing framework that enables the execution of computational tasks across multiple machines, allowing them to work together as a cohesive unit to process large datasets. These frameworks provide mechanisms for task distribution, fault tolerance, resource management, and inter-process communication. By abstracting the complexities of parallelism and machine coordination, these frameworks establish a separation of concerns between computation logic and Infrastructure Management.

6.3 Domain Application

Radio astronomy domain algorithms, encapsulated in the Common Astronomy Software Applications (CASA) API and library, are used to create Domain Applications that can be dynamically assigned to Technical Infrastructure compute resources at runtime. Multiple instances of the Domain Applications running in parallel on CPU- and GPU-based compute resources achieve horizonal and vertical scalability and satisfy the ngVLA system performance requirements. This arrangement leverages virtualization, container-based technologies, serverless cloud computing, and any future technology where applications should be agnostic regarding Technical Infrastructure.
Additionally, this arrangement allows domain experts to select any elements of the CASA API and library to create their own applications outside the context of the ngVLA system.

6.4 Infrastructure Management

The Infrastructure Management subsystem addresses the need to manage the execution of compute-intensive workflow tasks in Technical Infrastructure, and it does so by performing three principal functions[3]:

- resource allocation - assigning physical hardware, ranging from a fraction of a machine to an entire system, to processes based on user-configurable policies,
- workload scheduling - efficiently launching via optimized mechanisms thousands or more processes across a comparable number of nodes, and
- workload monitoring - overseeing application execution and tracking resource usage.

6.5 Subsystem Behaviors

Table 8 summarizes the SDP subsystems behaviors in terms of each subsystem's role in relation to one another based on a review of commercial off-the-shelf (COTS) and open-source technologies. Given CON7 (Python) and AD05 (containers), the table assumes for illustrative purposes the use of Prefect and Dask with Kubernetes, described further in Future Work, for a use case where:

- Infrastructure Management: Kubernetes provisions a cluster with multiple nodes and pods.

[3] Sterling et al., "High Performance Computing: Modern Systems and Practices", Morgan Kaufmann, 2025

- Parallel Compute Engine: Dask runs parallel image processing tasks across workers in the cluster.
- Workflow Management: Prefect schedules the overall workflow, including data staging, processing (delegated to Dask), and high-level product ingestion tasks.

Table 8. Subsystem Behaviors and Attributes.

Aspect	Workflow Management	Parallel Compute Engine	Infrastructure Management
Role	Workflow orchestration and task dependency management.	Parallel computing engine for Python.	Cluster and container orchestrator.
Primary Function	Manages workflows and execution timing.	Executes data-parallel Python computations.	Manages containerized workloads.
Granularity	High-level (flows, tasks, graphs)	Mid-level (tasks, datasets, graphs)	Low-level (pods, nodes, containers)
Scheduling Focus	Tasks across time and dependencies.	Tasks across workers.	Pods across nodes.
Fault Handling	Retries failed tasks, restarts flows.	Retries tasks, reassigns work.	Restarts failed pods, replicas.
User Interface	UI, CLI, APIs	UI, CLI, APIs	CLI, dashboards
Technical Infrastructure Awareness	Depends on Parallel Compute Engine and Resource Management (e.g., Dask, K8s).	Logical cluster via workers.	Full cluster/resource view.

Modern COTS and open-source Workflow Managers like Prefect and Airflow include adapters for supporting a variety of Parallel Compute Engines.

7 Conceptual Architecture Validation

In December 2024, the Radio Astronomy Data Processing Program initiated an agile prototyping effort to increase staff involvement in RADPS development and to validate the conceptual architecture[4,5]. A substantial body of work influenced the conceptual architecture and the resulting roadmap used to guide prototyping efforts. This section summarizes this work and concludes with the RADPS product roadmap.

7.1 Algorithm Architecture

"Algorithm Architecture v1.0" [RD11] presents a mathematical framework that establishes the basis for a set of components used for interferometric data processing. The components can combine in various ways to build iterative algorithms that satisfy imaging and calibration use cases. Some of the components relate abstractly to Workflow tasks and accomplish data preparation, data transform, and exit criteria steps. Similarly, other components relate to Infrastructure Management to accomplish parallel execution

[4] https://github.com/casangi/RADPS
[5] https://github.com/casangi/RADPS/wiki

steps. For example, an HTCImpl component might assign imaging subtasks across twenty nodes while each node might use HPCImpl to run GPU kernels or parallel CPU threads to process a subset of a larger dataset. These components are analogous to the Workload Management and Resource Management concepts, respectively.
Computationally intensive components described in [RD11] were deployed[6] using a set of C++ LibRA[7] applications as a pathfinder for ngVLA-scale processing. These applications mirror the structure and behavior of the Domain Application concept.

7.2 VIPER

"Dask Centered ngData Processing Architecture and VIPER Software Framework" [RD12] defines a modular and extensible Python-based architecture for scalable data reduction in radio astronomy, optimized for both performance via Dask and GPUs and usability via Jupyter Hub and reusable APIs. Figure 4 provides a layered view of the Dask-based parallel computing framework described in the document.

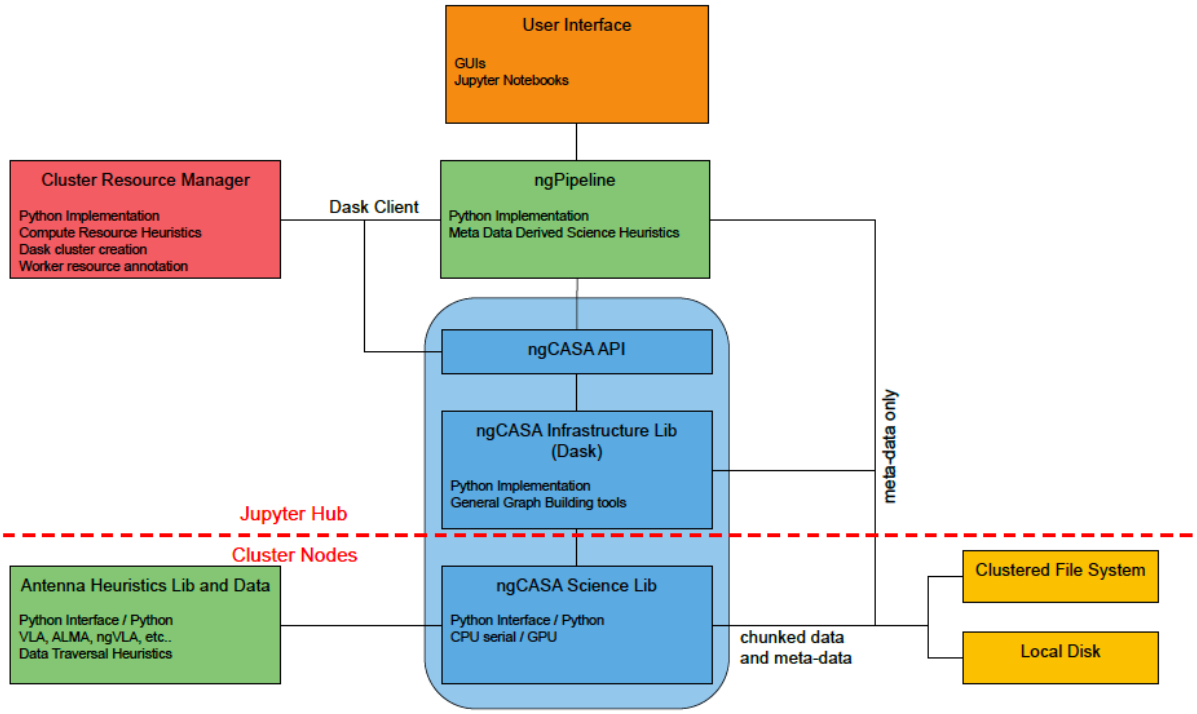

Figure 4. Dask-based layered parallel computing framework [RD12].

Analogies can be made between Figure 3 and Figure 4. For example, the User Interface corresponds to the SDP-UI concept and the ngCASA Science Lib and Antenna Heuristics Lib and Data boxes correspond to the Domain Application concept. Additionally, the ngPipeline and ngCASA API

[6] https://science.nrao.edu/enews/17.3/index.shtml#deepimaging
[7] https://github.com/ARDG-NRAO/LibRA

correspond to the Workflow Concept and the Cluster Resource Manager and ngCASA Infrastructure Lib map roughly to the Parallel Compute Engine concept.
Figure 5 shows a technology-centric view of the VIPER framework while retaining a sense of the different layers and highlighting the relationship between the processing framework and data sources. Many of the technologies shown in the figure are used in the prototyping described later.

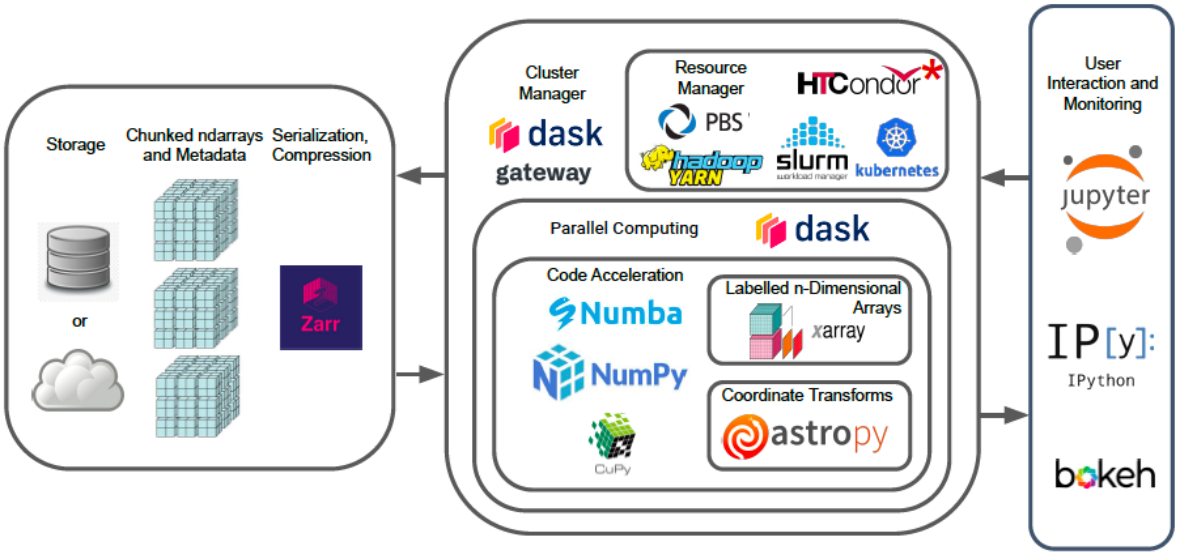

Figure 5. Technology view of the VIPER framework.

Figure 6 provides a VIPER-based activity diagram for an imaging use case that gives a sense of the iterative nature of the data processing along with a sense of the deployment for a Dask-based parallel computing framework.

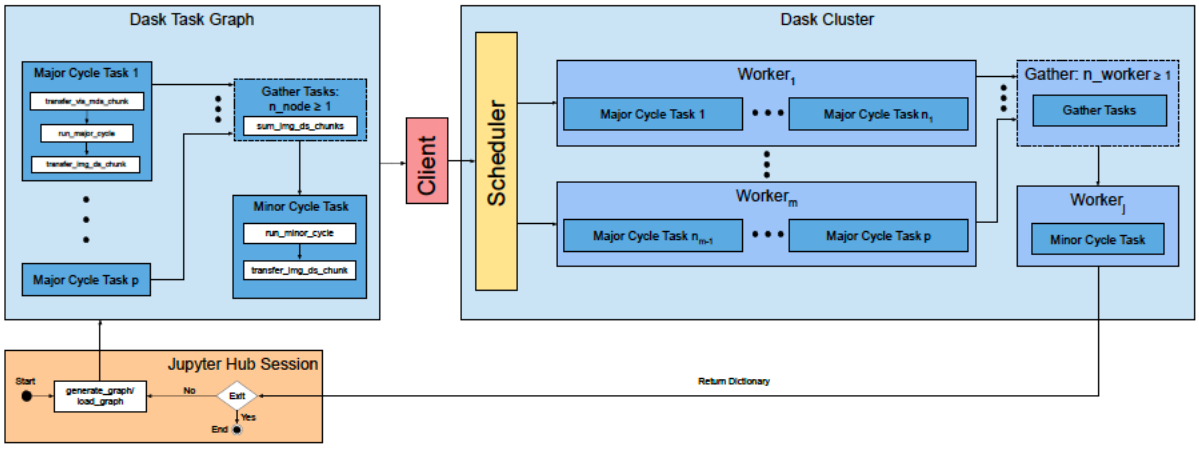

Figure 6. VIPER-based activity diagram for an imaging use case.

7.3 Example Pipeline Workflow

"An Example RADPS Workflow Decomposition" [RD13] presents a conceptual and technical breakdown of an end-to-end data reduction pipeline for radio astronomy within the context of the SDP conceptual architecture. The document offers concepts for a practical testbed that can be used to evaluate workflow decomposition, resource scheduling, parallelism strategies, checkpointing, compute reuse versus data transport trade-offs, and user interaction models.

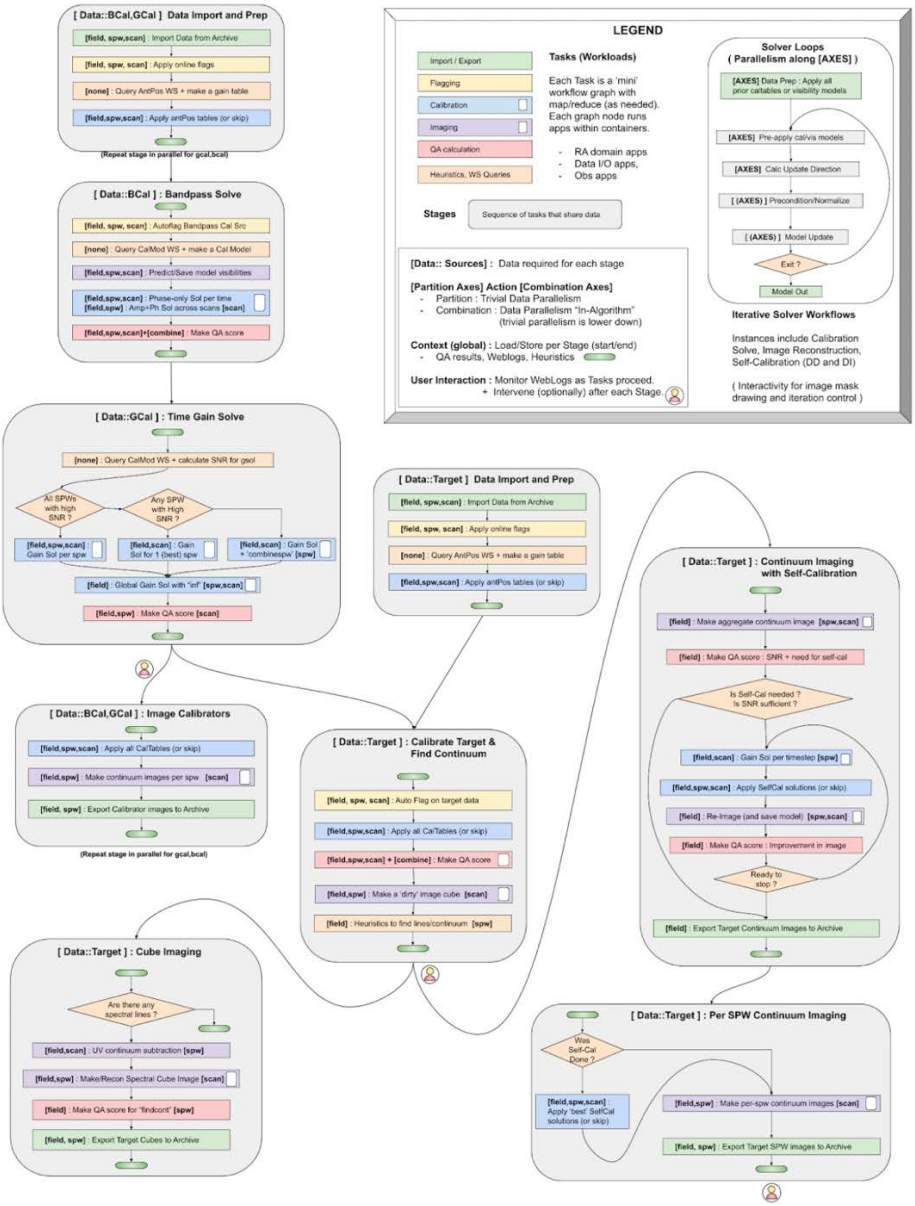

Figure 7. Example Workflow for end-to-end data reduction [RD13].

The stages and tasks show in Figure 7 correspond to the Workflow concept where some tasks are executed within the context of a workflow and other compute-intensive tasks are executed within the context of Infrastructure Management.

7.4 RADPS Product Roadmap

The roadmap shown in Table 9 defines the activities that will be used to validate the SDP conceptual architecture, support ALMA-WSU development, and advance the conceptual design to a preliminary design.

Table 9. DMS RADPS Prototyping Activities.

Activity	Description	Status
Concept Validation	Validate Workflow, Workflow Management, and Infrastructure Management concepts for Airflow and Prefect.	Completed
Cluster Infrastructure	Create an on-premises N node cluster with local and shared storage and a Kubernetes-based environment for Airflow, Prefect, and Dask.	Completed
Walking Skeleton - no data	Implement an example RADPS Workflow using Airflow, Prefect, and Dask executing in cluster infrastructure to validate end-to-end connections of Workflow Management, Parallel Execution Engine, and Infrastructure Management with a Jupyter Hub frontend.	Completed
Walking Skeleton - data	Implement an example RADPS Workflow using Airflow, Prefect, and Dask executing in cluster infrastructure and processing datasets described in NAASC Memo #121 [RD05].	In Progress
Scheduler Performance	Measure scheduler performance scaling in cluster infrastructure for Airflow, Prefect, and Dask.	In Progress
Domain Application Single Threaded Functions	Create imaging domain functions (ongoing, some functions for Walking Skeleton already exist).	In Progress
ALMA-WSU Demonstrator	Create and incorporate data processing and Domain Application execution into the walking skeleton.	Planned
ALMA-WSU Demonstrator Performance	Complete performance testing outlined in [RD03].	Planned
ALMA-WSU Demonstrator Cloud Deployment	Deploy Demonstrator on AWS and repeat performance testing. Optional.	Planned
ALMA-WSU Demonstrator Cluster Usage Efficiency	Calculate cluster usage efficiency.	Planned
Reliability Metrics	Identify Workflow Management and Infrastructure Management reliability metrics and describe how these metrics can be measured for test plan development.	Planned

8 Future Work

The strategy to advance the design throughout the preliminary design phase consists of two main efforts: DMS prototyping and external expert engagements. Other future work needed before PDR is also listed.

8.1 Prototyping

The first effort focuses DMS resources on NRAO's areas of expertise in interferometric and single dish data processing to build prototypes that validate the conceptual design and evaluate open source and COTS technologies. This effort is accomplished via Scrum teams developing increasingly realistic prototypes that can eventually be used to quantitatively measure various system attributes.
As an intermediate goal, this effort will deliver the ALMA-RADPS Demonstrator, "…publicly available code demonstrating the technologies that will be used for future RADPS-ALMA implementation for both the IF and TP use cases" [RD03]. Table 9 summarizes activities culminating in this goal.

8.2 External Expert Engagements

The second effort engages experts in the field of large-scale, distributed computing to provide feasible design options for on-premises, off-premises, cloud, and hybrid Technical Infrastructure as well as guidance for evolving the design, optimizing scalable radio astronomy algorithm performance, and achieving reliability requirements. As deliverables from this effort become available, they will be incorporated into the Scrum teams' product backlogs. The ngVLA project has established a contractual agreement with the Texas Advanced Computing Center[8] and an informal engagement plan with CI Compass[9].

8.2.1 Texas Advanced Computing Center

The National Science Foundation (NSF) Leadership-Class Computing Facility (LCCF) at the Texas Advanced Computing Center (TACC) is a world leader in large-scale data storage systems, software services and infrastructure for distributed computing systems, and optimizing software to maximize performance on a variety of hardware resources. LCCF deliverables are structured in two phases spanning April 15, 2025 to April 2027.
Phase 1 will deliver in 2025 a document that presents high-level design options for future data processing capabilities. This includes evaluating different facility models - whether hosted on-site, externally, in the cloud, or through a hybrid approach. The document will also outline considerations related to data movement, system performance, and processing infrastructure along the lines of previous ngVLA analysis ([RD09], [RD10]). Additionally, it will explore options for efficient data storage and reliable backup strategies to support long-term operational needs.
Phase 2 aims to advance the data processing and archival storage systems from a conceptual framework to a preliminary design. Over a 24-month period, a structured series of activities will be planned and documented, forming the foundation for a product backlog that guides prototyping efforts by the Data Management and Software (DMS) team at NRAO. These prototyping activities are intended to refine and focus the conceptual options identified in Phase 1, ultimately resulting in a well-defined preliminary

[8] https://tacc.utexas.edu/
[9] https://ci-compass.org/

design. A key emphasis of this phase is active engagement and training of DMS staff. The scope of work will encompass the full range of computing infrastructure for the ngVLA, including data movement, storage, throughput, algorithm performance, and tiered storage architecture.

8.2.2 CI Compass

CI Compass supports the NSF Major Facilities cyberinfrastructure needs by connecting people, knowledge, and processes to enable collaboration and discovery across these facilities. The CI Compass-ngVLA Engagement Plan proposes two working groups made up of DMS staff and CI Compass experts. The goal for one working group is to identify and recommend strategies to ensure that data processing pipelines remain reliable and resilient when running on large-scale compute resources. The scope of this work encompasses failure detection and recovery techniques for both Python and C++ domain applications, application-level checkpointing versus automated re-processing of failed tasks, and reliability features provided by container orchestration and workflow management systems.
The ngVLA will produce raw observational data, derived science data products, associated metadata, and operational/cyberinfrastructure data, all of which need to be stored, curated, and made accessible. The second working group will consider lessons learned from the ALMA observatory's data management and archive, as well as stakeholder requirements for the ngVLA archive, to identify and recommend data lifecycle management policies along with data integrity and replication strategies. The recommendations from this working group will complement the LCCF hardware designs related to data processing buffers and storage hierarchies.

8.3 Other Future Work

Table 10 lists other deliverables needed prior to PDR.

Table 10. Future work summary.

Activity	Deliverable
ALMA-D3 Interfaces	Use OpenAPI to define endpoints and AsyncAPI to define asynchronous, event-driven APIs between RADPS and ALMA-D3.
RADPS Behavior	Define RADPS interfaces and use them to create sequence diagrams that demonstrate RADPS behavior for various use cases.
Data Parallelism	Define mechanisms for data shuffling, partitioning, caching, or spilling.
Context Mechanism	The current "context" object used in the CASA pipeline system is a critical part of data flow, but its purpose, contents, and architectural appropriateness are poorly understood. There is a need to rethink, specify, and potentially redesign this mechanism within the scope of the ngVLA project.
Deployment Model	Incorporate a baseline deployment model into the design and document the influence of deployment options on SDP concepts.
Checkpointing	Evaluate through prototyping the feasibility of checkpointing as a reliability solution. Analysis should consider the bandwidth required for large numbers of processes to write to disk, requiring large bandwidth to the filesystem.
Off-premises UX	Determine interaction options related to launching and managing jobs on remote clusters.

Activity	Deliverable
Data Reports	Specify and design a replacement for web logs.
Test Plan Development	Define testable statements related to performance and reliability to support test plan development.
Technology Evaluation	Evaluate Workflow Management, Parallel Compute Engine, Infrastructure Management, and Domain Application technologies.
Cybersecurity	Refactor the design as needed to satisfy ngVLA cybersecurity requirements.
Python-C++ Interface	Evaluate the long-term performance implications of using pybind11 for Python bindings; periodically reassess its suitability compared to alternatives (e.g., nanobind) to ensure sustained efficiency over the ngVLA software lifecycle.
Infrastructure Management	Implement integrated infrastructure health and telemetry monitoring across all computational systems, linked with the workload manager to support automated triage, maintain production resource stability, and reduce operational workload failures.

9 Appendix A – Acronyms

Acronym	Definition
AD	Architecture Decision
ALMA	Atacama Large Millimeter/submillimeter Array
ASR	Architecturally Significant Requirement
CASA	Common Astronomy Software Package
COTS	Commercial Off The Shelf
CSS	Computing and Software System
DAG	Directed Acyclic Graph
DMS	Data Management and Software
GOUS	Group Observing Units Sets
HPC	High Performance Computing
IF	Interferometer or Interferometric
MOUS	Member Observing Units Sets
LCCF	Leadership-Class Computing Facility
MPI	Message Passing Interface
ngVLA	The Next Generation Very Large Array Project
NRAO	National Radio Astronomy Observatory
NSF	National Science Foundation
ONL	Online Data Acquisition System
PMN	Proposal Management System
QA	Quality Attribute
QAS	Quality Attribute Scenario
RADPS	Radio Astronomy Data Processing System
RFI	Radio Frequency Interference
SDA	Science Data Archive
SIT	Science Interface and Tools
SDP	Science Data Processor
TACC	Texas Advanced Computing Center
TI	Technical Infrastructure
TP	Total Power
WSU	Wideband Sensitivity Upgrade
VLA	Very Large Array
VLBA	Very Long Baseline Array